Abstract

The increasing disparity between processor speeds and main memory access times is a major limiting factor to program performance Caches memories are small, high speed buffer memories introduced between the processor and the main memory, which hold the most recently used code and data thus reducing this performance bottleneck Direct mapped caches are preferred as first level caches as they have the least access time The performance of a direct mapped of a given size, largely depends on the block size of the cache Small block sizes result in lower memory traffic, but need substantially large tag space Large block sizes have higher memory traffic owing to a large number of conflict misses and the implicit prefetching of data Existing solutions to reduce tag space and miss ratios in direct mapped caches led to design of subblock caches and decoupled sectored caches However, both these suffer from large number of conflicts and large tag space (especially with processors moving to 64 bit addresses) Our goal is to propose a cache design to reduce the miss ratios, memory traffic, tag space requirements without compromising on the access time of a direct mapped cache

In this thesis, we evaluate a new scheme to emulate a variable block size cache The scheme uses a subtagged cache (proposed in this thesis) in conjunction with prefetching In a subtagged cache, a portion of the tag of a cache block is associated with each subblock in the block This allows subblocks from different memory blocks to co-reside in the cache block at the same time This exploits temporal locality by reducing false conflicts at the block level To exploit spatial locality, we determine access patterns within cache blocks and prefetch multiple subblocks on a cache miss There is a significant variation in performance of the new schemes across platforms In particular, cache-block level conflicts between areas of the address space contribute significantly to miss ratios, even

when we use subtagging We observed that significant performance improvement occurs if the program stack is remapped so as to reduce the number of conflicting tag bits in a cache conflict and storing the conflicting bits with the subblock Overall, on suitable machine-compiler platforms where data regions are allocated in a cache-conscious fashion, we find that the new schemes for emulating variable block sizes outperform previously proposed alternatives, both in terms of miss ratios and memory traffic